What Is Claimed Is:

1. A method for executing a start transactional execution (STE) instruction to facilitate transactional execution on a processor, comprising:

encountering the STE instruction during execution of a program, wherein the STE instruction marks the beginning of a block of instructions to be executed transactionally; and

upon encountering the STE instruction, commencing transactional execution of the block of instructions following the STE instruction;

wherein changes made during the transactional execution are not committed to the architectural state of the processor until the transactional execution successfully completes.

2. The method of claim 1, wherein the STE instruction specifies an action to take if transactional execution of the block of instructions fails.

3. The method of claim 2, wherein the action to take can include branching to a location specified by the STE instruction.

4. The method of claim 2, wherein the action to take can include acquiring a lock on the block of instructions.

5. The method of claim 2, wherein the action to take can include setting state information within the processor to indicate a failure during transactional execution of the block of instructions, thereby enabling other software executed by the processor to manage the failure.

6. The method of claim 1, wherein if the transactional execution completes without encountering an interfering data access from another process or other type of failure, the method further comprises:

atomically committing changes made during the transactional execution, and

resuming normal non-transactional execution.

7. The method of claim 1, wherein if an interfering data access from another process is encountered during the transactional execution, the method further comprises:

discarding changes made during the transactional execution; and

attempting to re-execute the block of instructions.

8. The method of claim 1, wherein potentially interfering data accesses from other processes are allowed to proceed during the transactional execution of the block of instructions.

9. The method of claim 1, wherein the block of instructions to be executed transactionally comprises a critical section.

10. The method of claim 1, wherein commencing transactional execution of the block of instructions involves:

saving the state of processor registers;

configuring the processor to mark cache lines during loads that take place during transactional execution;

configuring the processor to mark cache lines during stores that take place during transactional execution; and configuring the processor to continually monitor data references from other threads to detect interfering data references.

11. The method of claim 1, wherein the STE instruction is a native machine code instruction of the processor.

12. The method of claim 1, wherein the STE instruction is defined in a platform-independent programming language.

13. A computer system that supports a start transactional execution (STE) instruction to facilitate transactional execution, wherein the STE instruction marks the beginning of a block of instructions to be executed transactionally, the computer system comprising:

a processor; and

an execution mechanism within the processor;

wherein upon encountering the STE instruction, the execution mechanism is configured to commence transactional execution of the block of instructions following the STE instruction;

wherein changes made during the transactional execution are not committed to the architectural state of the processor until the transactional execution successfully completes.

14. The computer system of claim 13, wherein the STE instruction specifies an action to take if transactional execution of the block of instructions fails.

15. The computer system of claim 14, wherein the action to take can include branching to a location specified by the STE instruction.

16. The computer system of claim 14, wherein the action to take can include acquiring a lock on the block of instructions.

17. The computer system of claim 14, wherein the action to take can include setting state information within the processor to indicate a failure during transactional execution of the block of instructions, thereby enabling other software executed by the processor to manage the failure.

18. The computer system of claim 13, wherein if the transactional execution completes without encountering an interfering data access from another process or other type of failure, the execution mechanism is configured to:

atomically commit changes made during the transactional execution, and to

resume normal non-transactional execution.

19. The computer system of claim 13, wherein if an interfering data access from another process is encountered during the transactional execution, the execution mechanism is configured to:

discard changes made during the transactional execution; and to

attempt to re-execute the block of instructions.

20. The computer system of claim 13, wherein the computer system is configured to allow potentially interfering data accesses from other processes to proceed during the transactional execution of the block of instructions.

21. The computer system of claim 13, wherein the block of instructions to be executed transactionally comprises a critical section.

22. The computer system of claim 13, wherein while commencing transactional execution of the block of instructions, the execution mechanism is configured to:

save the state of processor registers;

configure the processor to mark cache lines during loads that take place during transactional execution;

configure the processor to mark cache lines during stores that take place during transactional execution; and to

configure the processor to continually monitor data references from other threads to detect interfering data references.

23. The computer system of claim 13, wherein the STE instruction is a native machine code instruction of the processor.

24. The computer system of claim 13, wherein the STE instruction is defined in a platform-independent programming language.

25. A computing means that supports a start transactional execution (STE) instruction to facilitate transactional execution, wherein the STE instruction marks the beginning of a block of instructions to be executed transactionally, comprising:

a processing means; and

an execution means within the processing means;

wherein upon encountering the STE instruction, the execution means is configured to commence transactional execution of the block of instructions following the STE instruction;

wherein changes made during the transactional execution are not committed to the architectural state of the processor until the transactional execution successfully completes.